ITM1010
Computer and Communication Technologies

Lecture #3

Part I: Introduction to Computer Technologies

Numbering systems, Codes
and Boolean Algebra

Ones Complement

- The 1s complement of a binary number of a binary number is derived by subtracting each bit in the number to be complemented from 1.
- e.g. 1s complement of 1100 is 0011

$$\begin{array}{r} 1\ 1\ 1\ 1 \\ -\ 1\ 1\ 0\ 0 \\ \hline 0\ 0\ 1\ 1 \end{array}$$ (1s comp)

Subtraction in 1s Complement

$$\begin{array}{r} 1110 \\ -0010 \\ \hline \end{array}$$

$14_{(10)}-2_{(10)}=12_{(10)}$

Step 1: $-0010 = 1101_{(1s\ comp)}$

Step 2:
$$\begin{array}{r} 1110 \\ +1101 \text{ (1s comp)} \\ \hline \textcircled{1}\,1011 \end{array}$$

Step 3: $\begin{array}{r} +\quad 1 \text{ (End-around carry)} \\ \hline \mathbf{1100\ (Difference)} \end{array}$

Carry = 1

→ positive result

$$\begin{array}{r} 0111 \\ -1001 \\ \hline \end{array}$$

$7_{(10)}-9_{(10)}= -2_{(10)}$

Step 1: $-1001 = 0110_{(1s\ comp)}$

Step 2:
$$\begin{array}{r} 0111 \\ +0110 \\ \hline \textcircled{0}1101 \end{array}$$

Step 3: $\begin{array}{r} +\quad 0 \\ \hline 1101 \text{ (Difference in 1s complement form)} \end{array}$

$1101 = \mathbf{-0010}$

2s Complement

- Twos (2s) Complement = 1s complement + 1.

```
  1011
 -0101
```

Step 1: $-0101 = 1010_{(1s\ comp)}$

$+\ \ 1$

$1011_{(2s\ comp)}$

Step 2:

```
  1011
+ 1011
```

Step 3: (1) **0110 (Difference)**

Carry ←

$11_{(10)}-5_{(10)}=6_{(10)}$

Carry = 1

→ positive result

```
  1000
 -1100
```

$-1100 = 0011_{(1s\ comp)}$

$+\ \ 1$

$0100_{(2s\ comp)}$

```
  1000
+ 0100
 (0) 1100
```

Carry ←

$8_{(10)}-12_{(10)}=-4_{(10)}$

Class Exercise

- Subtract the following binary number using 2s complement.

0 1 0 1	0 0 1 1
- 0 1 0 0	- 0 1 0 1

Sign Bit

- A single bit, usually the leftmost bit, may be used to distinguish positive and negative numbers. The meaning of the sign bit can be fixed arbitrarily. But normally,

  sign bit

  0 - positive number

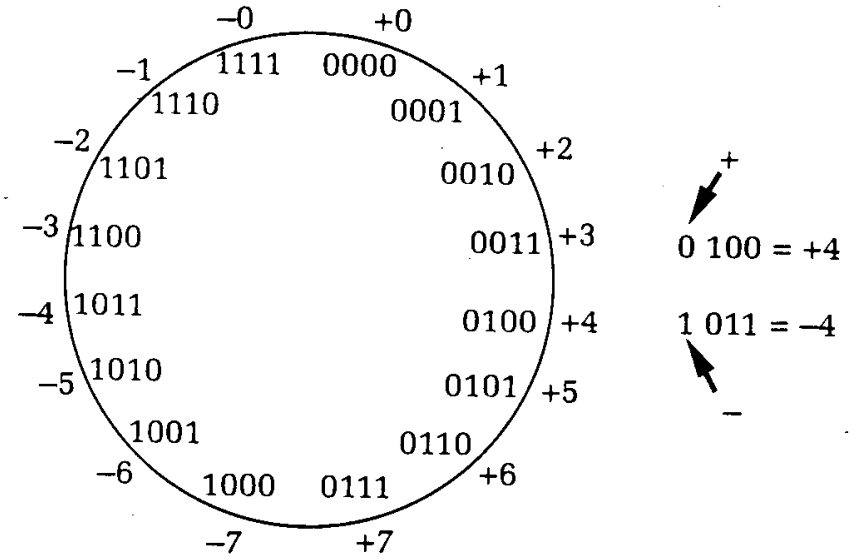
  1 - negative number

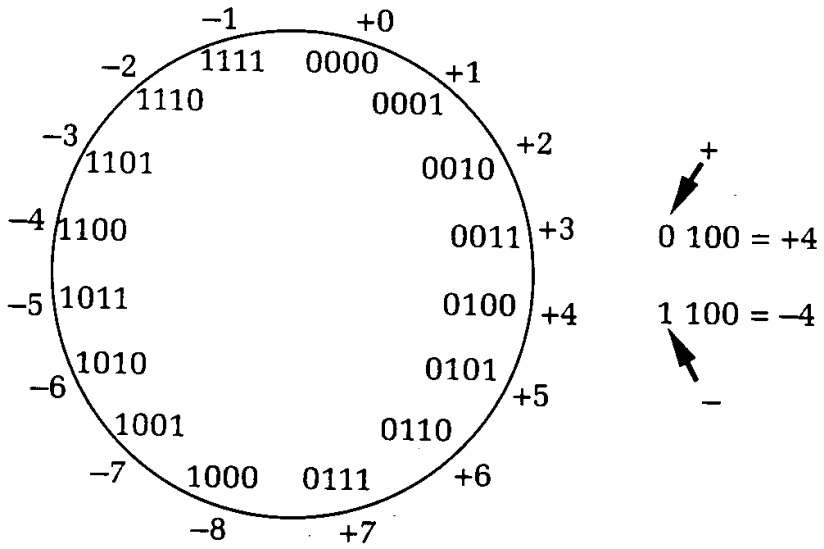
  e.g. $-5_{(10)} = 1101$

  $+5_{(10)} = 0101$

  - Note: the magnitude of a number is represented by the lower three bits

Sign Bit





The leftmost bit still indicates sign.

- In two's complement representation, two numbers can be added simply as two positive numbers e.g. 6 + (-2)= 4
- Overflow occurs whenever the sum of two positive numbers yields a negative result or when two negative numbers are summed and the result is positive.

Hexadecimal Numbering System

Hexadecimal is a convenient shorthand for a long string of binary digits: simply group the bits into sets of 4 and replace by the hex equivalent in order to convert from binary to hexadecimal. It has a base of 16. Example, $1001\ 1100\ 1111\ 0101_{(2)} = 9\ C\ F\ 5_{(16)}$

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Binary-Coded Decimal (BCD)

Decimal	Binary-Coded Decimal (BCD)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
–	1010*
–	1011*
–	1100*
–	1101*
–	1110*
–	1111*

Gray Code

Decimal	Binary	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

ASCII Codes

- American Standard Code for Information Interchange
- The ASCII encodes the letters in the alphabet as well as numbers, it is an alphanumeric code. It is a 7-bit code that allows representation of 128 different characters and commands, including
  - upper-case and lower-case letters
  - decimal numbers
  - punctuation marks
  - special symbols
  - command codes for formatting text
- Extended ASCII: 8-bit code allows for 128 additional graphics and international characters.

ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

Extended ASCII Codes

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	|	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	ÿ	168	¿	184	╕	201	╔	217	┘	233	Θ	249	∙
137	ë	153	Ö	169	⌐	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	ⁿ
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	φ	253	²
141	ì	158	₧	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

Source: www.asciitable.com

Boolean Algebra

Every Boolean expression has only one of the two possible answers: True or False.

Boolean Algebra - The Mathematics of Logic

- Boolean variables A, B, C, D, E etc. represent quantities which have only one of two possible values (binary); and
- Basic Operators on Boolean Variables

Product (AND) + Sum (OR) Complement (NOT)

$A \bullet B$ or AB $A+B$ $\overline{A}$

Basic Properties of Boolean Algebra

- Associative:

$$(A+B)+C=A+(B+C)=A+B+C$$

$$(AB)C=A(BC)=ABC$$

- Commutative:

$$A+B=B+A$$

$$AB=BA$$

- Distributive:

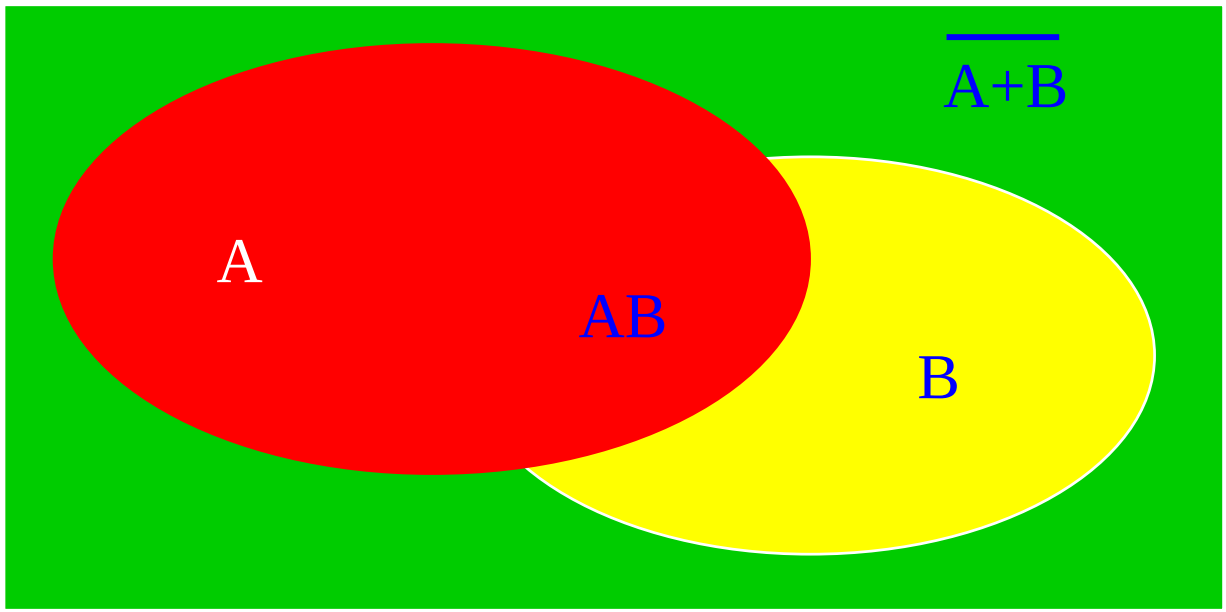
$$A(B+C)=AB+AC$$

Other Properties of Boolean Algebra

Property	Name
$A \cdot 0 = 0$	Inhibit
$A + 1 = 1$	Inhibit
$A \cdot 1 = A$	Enable
$A + 0 = A$	Enable
$A \cdot A = A$	Redundant
$A + A = A$	Redundant
$A \cdot \overline{A} = 0$	Complement
$A + \overline{A} = 1$	Complement
$A = \overline{\overline{A}}$	Double negation

Venn Diagram

- The Venn Diagram is made up of two or more overlapping circles. It is often used in mathematics to show relationships between sets.



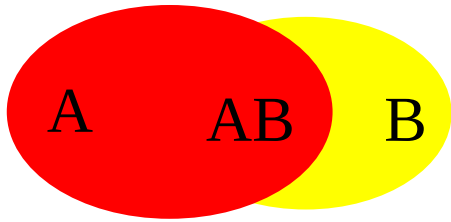
The Venn Diagram

- Property: A + AB = A

  Proof using

  Venn Diagram:

Venn diagram

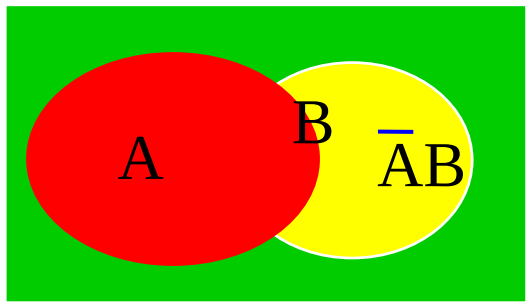


- Property: A + $\overline{A}$B = A+B

  Proof using

  Venn Diagram:

Venn diagram



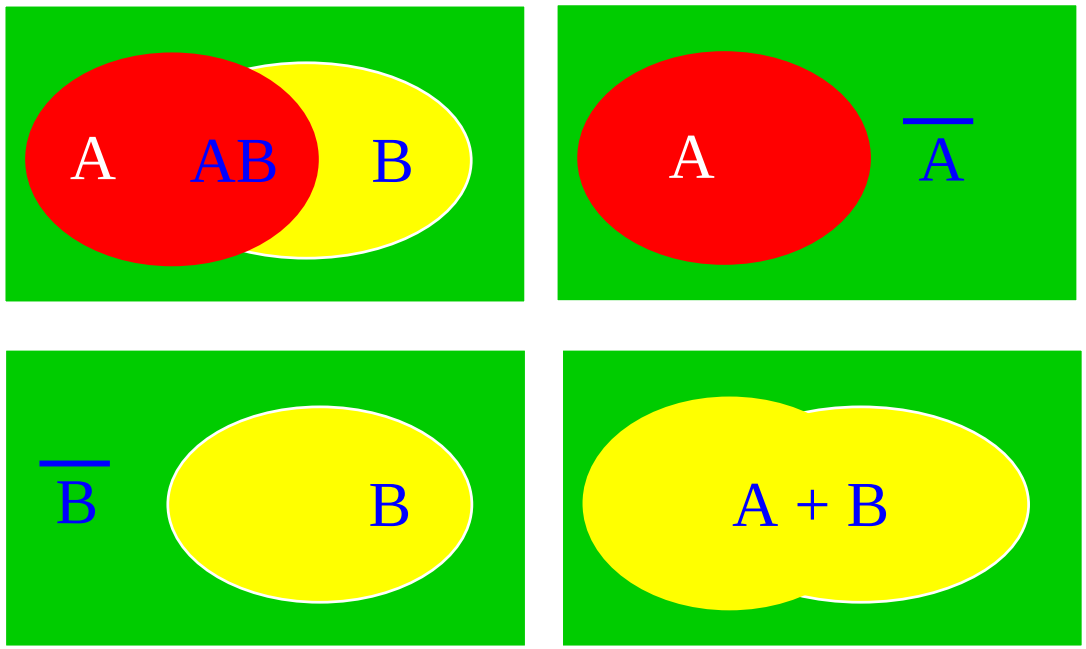
DeMorgan's Theorem

The complement of the SUM function is equal to the PRODUCT function of the complements.

$$\overline{A + B} = \overline{A}\,\overline{B} \quad <===> \quad \overline{AB} = \overline{A} + \overline{B}$$

$$\overline{A+B+C} = \overline{A}\,\overline{B}\,\overline{C}$$

$$\overline{ABC} = \overline{A}+\overline{B}+\overline{C}$$



Simplification of Expressions

$$(\overline{A}+B+\overline{C})(A+B+\overline{C})$$

$$=\overline{A}B+\overline{A}\,\overline{C}+BA+B+B\overline{C}+\overline{C}A+\overline{C}B+\overline{C}$$

$$=B(\overline{A}+A+1)+\overline{C}(A+B+1+\overline{A})$$

$$=B+\overline{C}$$

Truth Table

Truth table tabulates all possible results of an expression, e.g. Proof of DeMorgan's Theorem:

$$\overline{A + B} = \overline{A}\,\overline{B}$$

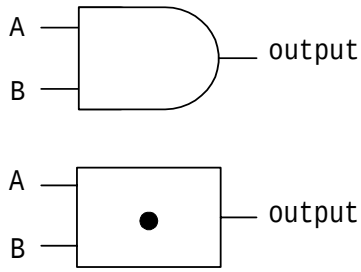
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

A	B	$\overline{A}\,\overline{B}$
0	0	1
0	1	0
1	0	0
1	1	0

Logic Gates - Building Blocks of Digital Circuits

- AND Gate

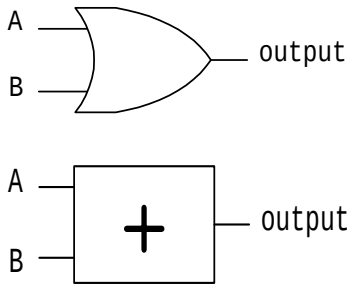
  Output = AB



A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

- OR Gate

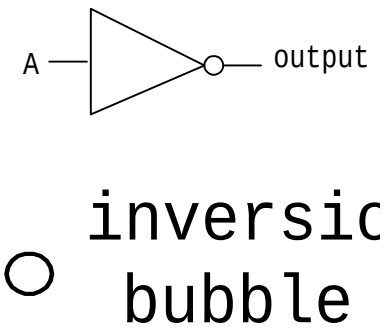
  Output = A + B



A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

- Inverter

  Output = A

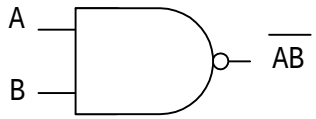


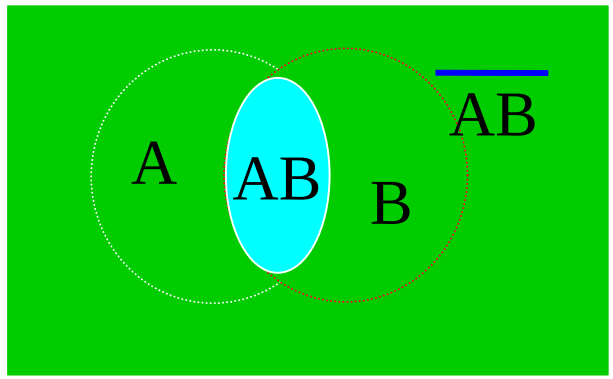
A	NOT A
0	1
1	0

Complete Set of Operations

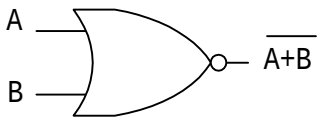
OR, AND & INVERTER together form a complete set: any Boolean function can be constructed from a combination of these three gates.

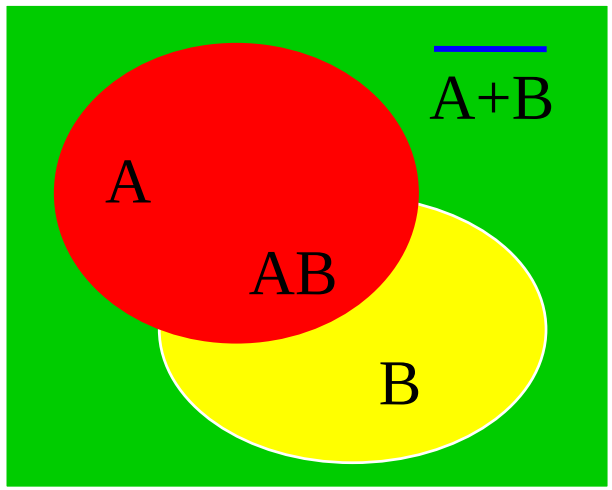
- NAND





- NOR



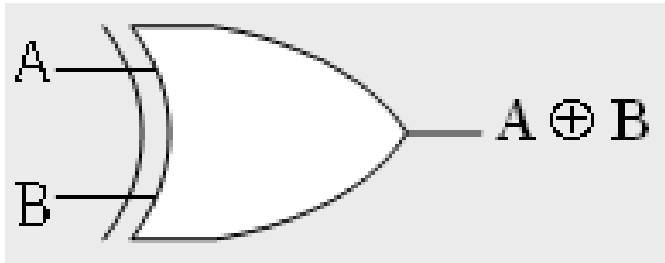


Yet Another Gate

Exclusive-OR (XOR):

This is useful as it is functionally equivalent to binary addition.

$$A \text{ XOR } B \equiv \bar{A}B + A\bar{B} = A \oplus B$$



A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

Properties:

- Commutative
- Associative
- Distributive

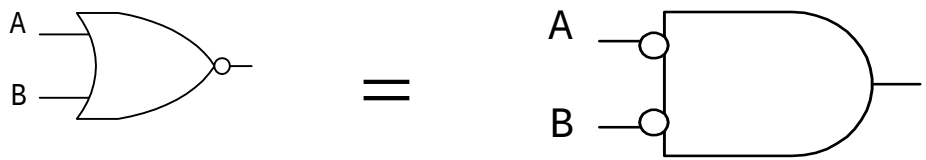
$$A \oplus B = B \oplus A$$

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$

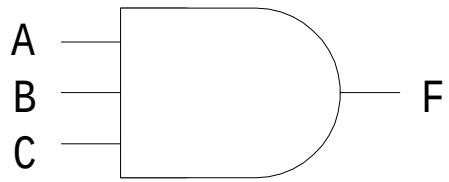
$$A(B \oplus C) = AB \oplus AC$$

De-Morgan's Theorem in Logic Gates

$$\overline{A + B} = \overline{A}\,\overline{B}$$



Logic Functions using Logic Gates



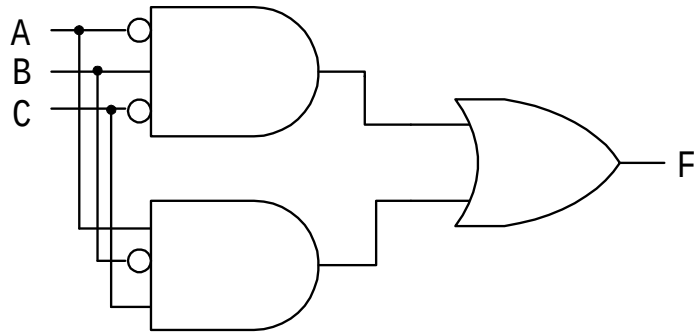
$$F = ABC$$

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

The term ABC can be written directly from the truth table as it corresponds with the binary pattern 111

Logic function implementation - SOP

This is usually called a sum-of-products (SOP) configuration.



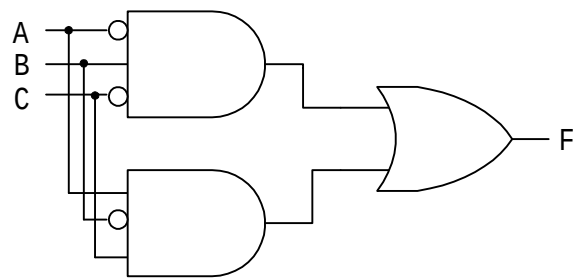
$$F = \overline{A}B\overline{C} + A\overline{B}C$$

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Product-Of-Sum (POS) Configuration

Product Of Sum obtained from truth table by making use of DeMorgan:

- OR (sum) the complemented inputs needed to get a low output in the truth table and AND (multiply) all such sums together



A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$$F = (A + B + C)(A + B + \overline{C})(A + \overline{B} + \overline{C})(\overline{A} + B + C)(\overline{A} + \overline{B} + C)(\overline{A} + \overline{B} + \overline{C})$$

Product-Of-Sum (POS) Configuration

In the POS extraction, each variable in a set of input variables that produces a low output is ORed together with other variables in that set. Afterward, each set of ORed variables that produce a low output is ANDed with other sets that produce low outputs. Extraction of the variables that produce low outputs is done in complementary form.

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$$F = (A + B + C)(A + B + \overline{C})(A + \overline{B} + \overline{C})(\overline{A} + B + C)(\overline{A} + \overline{B} + C)(\overline{A} + \overline{B} + \overline{C})$$

SOP Extraction vs. POS Extraction

- When an SOP expression is extracted from a truth table, the expression is written to represent each high output condition. This is because the OR gate will output high when any of the sets of input variables produces a high output from the AND gates.
- When a POS expression is extracted from a truth table, the expression is written to represent every low output condition on the truth table. This is because the AND gate will output low when any of the sets of input variables produce a low output from the OR gates.
- Using DeMorgan's theorem, expressions for SOP and POS are proved to be equal.

Some Definitions

- Minterm: product term containing all input variables of a function in either true or complementary form

  e.g. F= ABC

- Maxterm: sum term containing all input variables of a function in either true or complementary form

  e.g. F=A+B+C

- Canonical Form: a function expressed in either fully minterms or fully maxterms
- Literal: each occurrence of a variable of a function in either true or complementary form

Design Minimization

- Reduce Hardware
- Reduce Number of Inputs

May be realized in Boolean expression by having

- minimum number of terms
- minimum number of literals

Design Minimization using Boolean Algebra

Example

$$F = (A + B + C)(A + B + \overline{C})(A + \overline{B} + \overline{C})(\overline{A} + B + C)(\overline{A} + \overline{B} + C)(\overline{A} + \overline{B} + \overline{C})$$

no. of terms = 6

no. of literals = 18

The above expression may be simplified using Boolean algebra to:

$$F = \overline{A}B\overline{C} + A\overline{B}C$$